Evaluation of Chebyshev polynomials on intervals and application to root finding

Viviane Ledoux, Guillaume Moroz

Evaluation of Chebyshev polynomials on intervals and application to root finding

Viviane Ledoux[1,2] and Guillaume Moroz[1]

[1] INRIA, France `Firstname.Name@inria.fr`
[2] École Normale Supérieure `Firstname.Name@ens.fr`.

Abstract. In approximation theory, it is standard to approximate functions by polynomials expressed in the Chebyshev basis. Evaluating a polynomial f of degree n given in the Chebyshev basis can be done in $O(n)$ arithmetic operations using the Clenshaw algorithm. Unfortunately, the evaluation of f on an interval I using the Clenshaw algorithm with interval arithmetic returns an interval of width exponential in n. We describe a variant of the Clenshaw algorithm based on ball arithmetic that returns an interval of width quadratic in n for an interval of small enough width. As an application, our variant of the Clenshaw algorithm can be used to design an efficient root finding algorithm.

Keywords: Clenshaw algorithm · Chebyshev polynomials · Root finding · Ball arithmetic · Interval arithmetic.

1 Introduction

Clenshaw showed in 1955 that any polynomial given in the form

$$p(x) = \sum_{i=0}^{n} a_i T_i(x) \tag{1}$$

can be evaluated on a value x with a single loop using the following functions defined by recurrence:

$$u_k(x) = \begin{cases} 0 & \text{if } k = n+1 \\ a_n & \text{if } k = n \\ 2xu_{k+1}(x) - u_{k+2}(x) + a_k & \text{if } 1 \leq k < n \\ xu_1(x) - u_2(x) + a_0 & \text{if } k = 0 \end{cases} \tag{2}$$

such that $p(x) = u_0(x)$.

Unfortunately, if we use Equation (2) with interval arithmetic directly, the result can be an interval of size exponentially larger than the input, as illustrated in Example 1.

Example 1. Let $\varepsilon > 0$ be a positive real number, and let x be the interval $[\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon]$ of width 2ε. Assuming that $a_n = 1$, we can see that u_{n-1} is an interval of width 4ε. Then by recurrence, we observe that u_{n-k} is an interval of width at least $4\varepsilon F_k$ where $(F_n)_{n\in\mathbb{N}}$ denotes the Fibonacci sequence, even if all $a_i = 0$ for $i < n$.

Note that the constant below the exponent is even higher when x is closer to 1. These numerical instabilities also appear with floating point arithmetic near 1 and -1 as analyzed in [4].

To work around the numerical instabilities near 1 and -1, Reinsch suggested a variant of the Clenshaw algorithm [4,7]. Let $d_n(x) = a_n$ and $u_n(x) = a_n$, and for k between 0 and $n-1$, define $d_k(x)$ and $u_k(x)$ by recurrence as follows:

$$\begin{cases} d_k(x) = 2(x-1)u_{k+1}(x) + d_{k+1}(x) + a_k \\ u_k(x) = d_k(x) + u_{k+1} \end{cases} \tag{3}$$

Computing $p(x)$ with this recurrence is numerically more stable near 1. However, this algorithm does not solve the problem of exponential growth illustrated in Example 1.

Our first main result is a generalization of Equation 3 for any value in the interval $[-1,1]$. This leads to Algorithm 1 that returns intervals with tighter radii, as analyzed in Lemma 2. Our second main result is the use of classical backward error analysis to derive Algorithm 2 which gives an even better radii. Then in Section 3 we use the new evaluation algorithm to design a root solver for Chebyshev series, detailed in Algorithm 3.

2 Evaluation of Chebyshev polynomials on intervals

2.1 Forward error analysis

In this section we assume that we want to evaluate a Chebyshev polynomial on the interval I. Let a be the center of I and r be its radius. Furthermore, let γ and $\overline{\gamma}$ be the 2 conjugate complex roots of the equation:

$$X^2 - 2aX + 1 = 0. \tag{4}$$

In particular, using Vieta's formulas that relate the coefficients to the roots of a polynomial, γ satisfies $\gamma + \overline{\gamma} = 2a$ and $\gamma\overline{\gamma} = 1$.

Let $z_n(x) = a_n$ and $u_n(x) = a_n$, and for k between 0 and $n-1$, define $z_k(x)$ and $u_k(x)$ by recurrence as follows:

$$\begin{cases} z_k(x) = 2(x-a)u_{k+1}(x) + \gamma z_{k+1}(x) + a_k \\ u_k(x) = z_k(x) + \overline{\gamma} u_{k+1}(x) \end{cases} \tag{5}$$

Using Equation (4), we can check that the u_k satisfies the recurrence relation $u_k(x) = 2xu_{k+1}(x) - u_{k+2}(x) + a_k$, such that $p(x) = xu_1(x) - u_2(x) + a_0$.

Let (e_k) and (f_k) be two sequences of positive real numbers. Let $\mathcal{B}_{\mathbb{R}}(a,r)$ and $\mathcal{B}_{\mathbb{R}}(u_k(a), e_k)$ represent the intervals $[a-r, a+r]$ and $[u_k(a) - e_k, u_k(a) + e_k]$. Let $\mathcal{B}_{\mathbb{C}}(z_k(a), f_k)$ be the complex ball of center $z_k(a)$ and radius f_k.

Our goal is to compute recurrence formulas on the e_k and the f_k such that:

$$\begin{cases} z_k(\mathcal{B}_{\mathbb{R}}(a,r)) \subset \mathcal{B}_{\mathbb{C}}(z_k(a), f_k) \\ u_k(\mathcal{B}_{\mathbb{R}}(a,r)) \subset \mathcal{B}_{\mathbb{R}}(u_k(a), e_k). \end{cases} \tag{6}$$

Lemma 1. *Let $e_n = 0$ and $f_n = 0$ and for $n > k \geq 1$:*

$$\begin{cases} f_k = 2r|u_{k+1}(a)| + 2re_{k+1} + f_{k+1} \\ e_k = \min(e_{k+1} + f_k, \frac{f_k}{\sqrt{1-a^2}}) \text{ if } |a| < 1 \text{ else } e_{k+1} + f_k \end{cases} \tag{7}$$

Then, (e_k) and (f_k) satisfy Equation (6).

Proof (sketch). For the inclusion $z_k(\mathcal{B}_{\mathbb{R}}(a,r)) \subset \mathcal{B}_{\mathbb{C}}(z_k(a), f_k)$, note that γ has modulus 1, such that the radius of γz_{k+1} is the same as the radius of z_{k+1} when using ball arithmetics. The remaining terms bounding the radius of z_k follow from the standard rules of interval arithmetics.

For the inclusion $u_k(\mathcal{B}_{\mathbb{R}}(a,r)) \subset \mathcal{B}_{\mathbb{C}}(u_k(a), e_k)$, note that the error segment on u_k is included in the Minkowski sum of a disk of radius f_k and a segment of radius e_{k+1}, denoted by M. If θ is the angle of the segment with the horizontal, we have $\cos\theta = a$. We conclude that the intersection of M with a horizontal line is a segment of radius at most $\min(e_{k+1} + f_k, \frac{f_k}{\sqrt{1-a^2}})$.

Corollary 1. *Let $\mathcal{B}_{\mathbb{R}}(u,e) =$ `BallClenshawForward`$((a_0, \ldots, a_n), a, r)$ be the result of Algorithm 1, then*

$$p(\mathcal{B}_{\mathbb{R}}(a,r)) \subset \mathcal{B}_{\mathbb{R}}(u,e)$$

Moreover, the following lemma bounds the radius of the ball returned by Algorithm 1.

Lemma 2. *Let $\mathcal{B}_{\mathbb{R}}(u,e) =$ `BallClenshawForward`$((a_0, \ldots, a_n), a, r)$ be the result of Algorithm 1, and let M be an upper bound on $|u_k(a)|$ for $1 \leq k \leq n$. Assume that $\varepsilon_k < Mr$ for $1 \leq k \leq n$, then*

$$\begin{cases} e < 2Mn^2r & \text{if} \quad n < \frac{1}{2\sqrt{1-a^2}} \\ e < 9Mn\frac{r}{\sqrt{1-a^2}} & \text{if} \quad \frac{1}{2\sqrt{1-a^2}} \leq n < \frac{\sqrt{1-a^2}}{2r} \\ e < 2M\left[(1+\frac{2r}{\sqrt{1-a^2}})^n - 1\right] & \text{if} \quad \frac{\sqrt{1-a^2}}{2r} < n \end{cases}$$

Proof (sketch). We distinguish 2 cases. First if $n < \frac{1}{2\sqrt{1-a^2}}$, we focus on the relation $e_k \leq e_{k+1} + f_k + Mr$, and we prove by descending recurrence that $e_k \leq 2M(n-k)^2r$ and $f_k \leq 2Mr(2(n-k-1)+1)$.

For the case $\frac{1}{2\sqrt{1-a^2}} \leq n$, we use the relation $e_k \leq \frac{f_k}{\sqrt{1-a^2}} + Mr$, that we substitute in the recurrence relation defining f_k to get $f_k \leq 2rM + \frac{2r}{\sqrt{1-a^2}} f_{k+1} + f_{k+1} + Mr\sqrt{1-a^2}$. We can check by recurrence that $f_k \leq \frac{3}{2}M\sqrt{1-a^2}\left[(1+\frac{2r}{\sqrt{1-a^2}})^n - 1\right]$, which allows us to conclude for the case $\frac{\sqrt{1-a^2}}{2r} \leq n$. Finally, when $\frac{1}{2\sqrt{1-a^2}} \leq n < \frac{\sqrt{1-a^2}}{2r}$, we observe that $(1+\frac{2r}{\sqrt{1-a^2}})^n - 1 \leq n\exp(1)\frac{2r}{\sqrt{1-a^2}}$ which leads to the bound for the last case.

Algorithm 1 Clenshaw evaluation algorithm, forward error

function BallClenshawForward($(a_0, \ldots, a_n)$, a, r)
  ▷ *Computation of the centers* u_k
  $u_{n+1} \leftarrow 0$
  $u_n \leftarrow a_n$
  for k in $n-1, n-2, \ldots, 1$ **do**
    $u_k \leftarrow 2au_{k+1} - u_{k+2} + a_k$
    $\varepsilon_k \leftarrow$ bound on the rounding error for u_k
  $u_0 \leftarrow au_1 - u_2 + a_0$
  $\varepsilon_0 \leftarrow$ bound on the rounding error for u_0

  ▷ *Computation of the radii* e_k
  $f_n \leftarrow 0$
  $e_n \leftarrow 0$
  for k in $n-1, n-2, \ldots, 1$ **do**
    $f_k \leftarrow 2r|u_{k+1}| + 2re_{k+1} + f_{k+1}$
    $e_k \leftarrow min(e_{k+1} + f_k, \frac{f_k}{\sqrt{1-a^2}}) + \varepsilon_k$
  $f_0 \leftarrow r|u_1| + 2re_1 + f_1$
  $e_0 \leftarrow min(e_1 + f_0, \frac{f_0}{\sqrt{1-a^2}}) + \varepsilon_0$
  return $\mathcal{B}_{\mathbb{R}}(u_0, e_0)$

2.2 Backward error analysis

In the literature, we can find an error analysis of the Clenshaw algorithm [3]. The main idea is to add the errors appearing at each step of the Clenshaw algorithm to the input coefficients. Thus the approximate result correspond to the exact result of an approximate input. Finally, the error bound is obtained as the evaluation of a Chebyshev polynomial. This error analysis can be used directly to derive an algorithm to evaluate a polynomial in the Chebyshev basis on an interval in Algorithm 2.

Lemma 3. *Let* $e_n = 0$ *and for* $n > k \geq 1$*:*

$$e_k = 2r|u_{k+1}(a)| + e_{k+1} \tag{8}$$

and $e_0 = r|u_1(a)| + e_1$*. Then* (e_k) *satisfies* $u_k(\mathcal{B}_{\mathbb{R}}(a, r)) \subset \mathcal{B}_{\mathbb{R}}(u_k(a), e_k)$*.*

Proof (sketch). In the case where the computations are performed without errors, D. Elliott [3, Equation (4.9)] showed that for $\gamma = \tilde{x} - x$ we have:

$$p(\tilde{x}) - p(x) = 2\gamma \sum_{i=0}^{n} u_i(\tilde{x}) T_i(x) - \gamma u_1(\tilde{x})$$

In the case where $\tilde{x} = a$ and $x \in \mathcal{B}_{\mathbb{R}}(a, r)$ we have $\gamma \leq r$ and $|T(x)| \leq 1$ which implies $e_k \leq r|u_1(a)| + \sum_{i=2}^{n} 2r|u_i(a)|$.

Corollary 2. *Let* $\mathcal{B}_{\mathbb{R}}(u,e) = \texttt{BallClenshawBackward}((a_0,\ldots,a_n),a,r)$ *be the result of Algorithm 2, and let* M *be an upper bound on* $|u_k(a)|$ *for* $1 \le k \le n$. *Assume that* $\varepsilon_k < Mr$ *for* $1 \le k \le n$, *then* $e < 3Mnr$.

Algorithm 2 Clenshaw evaluation algorithm, backward error

function BallClenshawBackward($(a_0,\ldots,a_n)$, a, r)
- ▷ *Computation of the centers* u_k
- $u_{n+1} \leftarrow 0$
- $u_n \leftarrow a_n$
- **for** k in $n-1, n-2, \ldots, 1$ **do**
  - $u_k \leftarrow 2au_{k+1} - u_{k+2} + a_k$
  - $\varepsilon_k \leftarrow$ bound on the rounding error for u_k
- $u_0 \leftarrow au_1 - u_2 + a_0$
- $\varepsilon_0 \leftarrow$ bound on the rounding error for u_0
- ▷ *Computation of the radii* e_k
- $e_n \leftarrow 0$
- **for** k in $n-1, n-2, \ldots, 1$ **do**
  - $e_k \leftarrow e_{k+1} + 2r|u_{k+1}| + \varepsilon_k$
- $e_0 \leftarrow e_1 + r|u_1| + \varepsilon_0$
- **return** $\mathcal{B}_{\mathbb{R}}(u_0, e_0)$

3 Application to root finding

For classical polynomials, numerous solvers exist in the literature, such as those described in [5] for example. For polynomials in the Chebyshev basis, several approaches exist that reduce the problem to polynomial complex root finding [1], or complex eigenvalue computations [2] among other.

In this section, we experiment a direct subdivision algorithm based on interval evaluation, detailed in Algorithm 3. This algorithm is implemented and publicly available in the software `clenshaw` [6].

We applied this approach to Chebyshev polynomials whose coefficients are independently and identically distributed with the normal distribution with mean 0 and variance 1.

As illustrated in Figure 1 our code performs significantly better than the classical companion matrix approach. In particular, we could solve polynomials of degree 90000 in the Chebyshev basis in less than 5 seconds and polynomials of degree 5000 in 0.043 seconds on a quad-core Intel(R) i7-8650U cpu at 1.9GHz. For comparison, the standard numpy function `chebroots` took more than 65 seconds for polynomials of degree 5000. Moreover, using least square fitting on the ten last values, we observe that our approach has an experimental complexity closer to $\Theta(n^{1.67})$, whereas the companion matrix approach has a complexity closer to $\Theta(n^{2.39})$.

Algorithm 3 Subdivision algorithm for root finding

Require: $(a_0, \ldots, a_n)$ represents the Chebyshev polynomial approximating $f(x)$
$(b_0, \ldots, b_n)$ represents the Chebyshev polynomial approximating $\frac{df}{dx}(x)$
Ensure: *Res* is a list of isolating intervals for the roots of f in $[-1, 1]$

function SubdivideClenshaw($(a_0, \ldots, a_n)$, $(b_0, \ldots, b_n)$)
  ▷ *Partition* $[-1, 1]$ *in intervals where* F *either has constant sign or is monotonous*
  $L \leftarrow [\mathcal{B}_{\mathbb{R}}(0, 1)]$
  $Partition \leftarrow []$
  while L is not empty **do**
    $\mathcal{B}_{\mathbb{R}}(a, r) \leftarrow$ `pop` the first element of L
    $\mathcal{B}_{\mathbb{R}}(f, s) \leftarrow$ `BallClenshaw` $((a_0, \ldots, a_n), a, r)$
    $\mathcal{B}_{\mathbb{R}}(df, t) \leftarrow$ `BallClenshaw` $((b_0, \ldots, b_n), a, r)$
    if $f - s > 0$ **then**
      `append` the pair $(\mathcal{B}_{\mathbb{R}}(a, r), \text{"}plus\text{"})$ to *Partition*
    else if $f + s < 0$ **then**
      `append` the pair $(\mathcal{B}_{\mathbb{R}}(a, r), \text{"}minus\text{"})$ to *Partition*
    else if $g - s > 0$ or $g + s < 0$ **then**
      `append` the pair $(\mathcal{B}_{\mathbb{R}}(a, r), \text{"}monotonous\text{"})$ to *Partition*
    else
      $\mathcal{B}_1, \mathcal{B}_2 \leftarrow$ `subdivide`$\mathcal{B}_{\mathbb{R}}(a, r)$
      `append` $\mathcal{B}_1, \mathcal{B}_2$ to L

  ▷ *Compute the sign of* F *at the boundaries*
  $\mathcal{B}_{\mathbb{R}}(f, s) \leftarrow$ `BallClenshaw` $((a_0, \ldots, a_n), -1, 0)$
  `append` the pair $(\mathcal{B}_{\mathbb{R}}(-1, 0), \texttt{sign}(\mathcal{B}_{\mathbb{R}}(f, s)))$ to *Partition*
  $\mathcal{B}_{\mathbb{R}}(f, s) \leftarrow$ `BallClenshaw` $((a_0, \ldots, a_n), 1, 0)$
  `append` the pair $(\mathcal{B}_{\mathbb{R}}(1, 0), \texttt{sign}(\mathcal{B}_{\mathbb{R}}(f, s)))$ to *Partition*

  ▷ *Recover the root isolating intervals*
  $Partition \leftarrow$ `sort` *Partition*
  $Res \leftarrow$ the "monotonous" intervals of *Partition*
    such that the adjacent intervals have opposite signs
  return *Res*

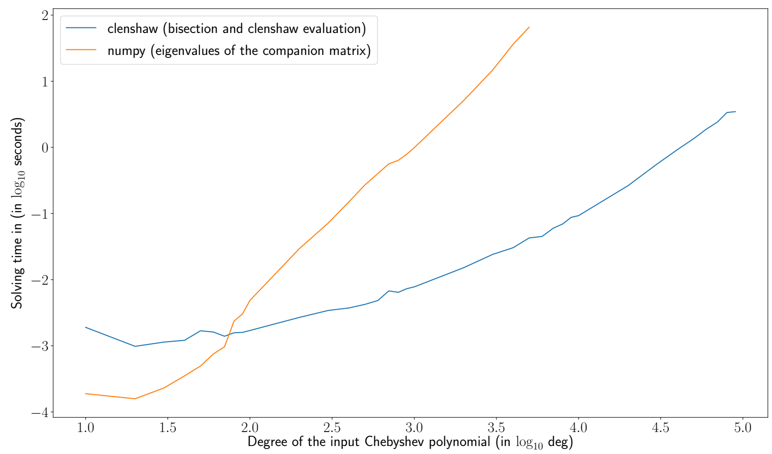

Fig. 1. Time for isolating the roots of a random Chebyshev polynomial, on a quad-core Intel(R) i7-8650U cpu at 1.9GHz, with 16G of ram

References

1. Boyd, J.: Computing zeros on a real interval through chebyshev expansion and polynomial rootfinding. SIAM Journal on Numerical Analysis **40**(5), 1666–1682 (2002). https://doi.org/10.1137/S0036142901398325
2. Boyd, J.: Finding the zeros of a univariate equation: Proxy rootfinders, chebyshev interpolation, and the companion matrix. SIAM Review **55**(2), 375–396 (2013). https://doi.org/10.1137/110838297
3. Elliott, D.: Error analysis of an algorithm for summing certain finite series. Journal of the Australian Mathematical Society **8**(2), 213–221 (1968). https://doi.org/10.1017/S1446788700005267
4. Gentleman, W.M.: An error analysis of Goertzel's (Watt's) method for computing Fourier coefficients. The Computer Journal **12**(2), 160–164 (01 1969). https://doi.org/10.1093/comjnl/12.2.160
5. Kobel, A., Rouillier, F., Sagraloff, M.: Computing real roots of real polynomials ... and now for real! In: Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation. pp. 303–310. ISSAC '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2930889.2930937
6. Moroz, G.: Clenshaw 0.1 (Dec 2019). https://doi.org/10.5281/zenodo.3571248, `https://gitlab.inria.fr/gmoro/clenshaw`
7. Oliver, J.: An Error Analysis of the Modified Clenshaw Method for Evaluating Chebyshev and Fourier Series. IMA Journal of Applied Mathematics **20**(3), 379–391 (11 1977). https://doi.org/10.1093/imamat/20.3.379